Postel

RFC 285   NIC 8271

Network Working Group
Request for Comments:285
NIC:8271
Updates: None
Obsoletes: None

Donald Huff
CWRU (Case)
December 15, 1971

Rcvd 4 JAN72

NETWORK GRAPHICS

Not much has been written about graphics on the ARPANET when the volume of the NIC collection is considered. Presently it contains some 8000 entries of which only about 20 are on the subject of graphics. The reason is probably similar to that given by L.G. Roberts in A FORWARD LOOK (NIC 7542) as the reason that data base sharing or software sharing will not be important topics for several more years, the NET hasn't been up long enough for interested people to have enough of the facts to know if it is feasible and to think creatively.

This paper is therefore aimed at bringing together the present state of graphics on the NET for the newcomer and attempting to add a little more distance to the ground covered so far. I will start with an over view, then proceed to briefly describe past work, and finally add some of my own thoughts.

Since the NET represents a wealth of data processors, any or all of which may be used at one time, we are not restricted to the configurations most generally found in private installations where there is a main processor and a somewhat less capable machine or perhaps none at all doing the honors as display processor. Indeed when using the NET it might occur that one has a more powerful machine as a display processor than the machine which is running the main job. Graphics on the NET need not be anything like what we know it as now.

There is of course a greater more diversified mix of graphics equipment that must be considered when designing a standard graphics language and its processor. If we wish to drive an arbirtary display from a program such an output language must be quite general, but the processor which constructs the actual display list for the target display need not and infact will not be general, rather its only job will be to translate a well defined general language to meet the requirements of one specific graphics terminal.

Attention handling, a lately discussed and much worried about topic, presents an entirely different problem. This time the NET may cause more harm than good for the simple reason that now there may be several, instead of one (in some cases none at

all), mappings defined to get from the initial display list that the main job process is creating to the final display list which interactive devices such as the light pen actually refer to. This is a problem which has to be faced and has been solved at many different sites in as many different ways. It is likely to give as much trouble as the final concept.

Local processing is in many cases a very simple thing to accomplish when the display terminal is "intellegent" or even has its own medium or large scale processor which has little or nothing else to do aside from refreshing the display. Such processing can be simple additions or deletions to the picture which certainly do not require the main job process to acomplish. The local processor need only notify the main process of what changes have been made to the display list so that the main copy may be updated. The allocation of abilities poses the last problem. The lower limit is reached when the local processor is unable to do anything beyond keeping the picture displayed, and the upper limit applies to the case when the local processor is more powerful than the main processor and handles all attentions itself. Now such questions as, just which copy of the display list is the master copy, who is responsible for seeing that all copies of the list contain the same information, and what kind of mappings between display lists are required, become the important

ones we all seek to answer.

Proposals for Network Standard Graphics started with the idea of a simple interpretable language containing only commands to erase the screen, display a string of text, move the beam or draw a line or point within a virtual rectangle which is the generalized display screen, execute a previously defined subroutine, and replace the contents of a subroutine with what follows in the command stream. Movements within the screen area were defined in terms of fractions of the screen dimensions instead of absolute lengths. This proposal was responded to with the suggestion that a graphics standard could not be so restrictive and find wide acceptence. The proposal was not expressive enough to handle sophisticated picture manipulations. It was recognized that a standard must be able to make use of all graphics hardware, present and within the forseeable future. The data structure should represent both logical and pictorial structure, allow for the definition and manipulation of subpictures, and division of the display screen into logical units. The proposed standard has now become a general high-level language rather than a low-level language. It was pointed out that all sites need not be able to handle the interpretation of this graphic language, but because of the existance of the rest of the NET, one of the other machines could run the interpreter.

this is equivalent to a data reconfiguration service. Such drawing modes as intensity, blinking, dashed, color, or stereo should also be expressable by means of a command to set the mode. The canonical definition of a character string should be defined since everyone has there own way of displaying text and most of them are different. It is suggested that the multics convention be used as described by Osanna, J., Sahzer, J., Remote Terminal Character Stream Processing of Multics, Proceedings SJCC, 1970, p. 671.

If in addition to simply displaying graphic information, if one wishes to interact with the picture directly, the protocol must include a standard for feedback, attention handling as it is being called. Attentions may not always refer directly to the picture however, as in the case of keyboard input which can be handled as any other standard message on the NET. Some graphics processors may also have the capability of handling attentions locally and only need to report the end result to the main process. This is the problem of which data structure is most up to date, which is considered the master copy, and how can the processes be kept in sync? The observation is also made that as long as the graphics application program, the main process, communicates with a pair of graphic device handling routines in a network standard language, the system configuration can be

arbitrary and any terminal may be attached to any main process. The same is of course true of attention handling, a set of standards for the transmittion of an attention generated by a particular device when developed will allow any graphics terminal to be understood by any other main process. A summary of input devices has been given along with typical outputs and the suggestion that each attention message identify the device causing the attention, the data which is being supplied, and of course, the data itself.

The proposed graphic protocol has become much richer in display types. The following list was suggested as basic: points, lines, vectors, character strings, viewport and window, transformations of instances, hardware-dependent byte streams, attention commands. The point was also made that special considerations for grey-scale devices are needed and four alternate display modes are discussed (NIC 7128).

An example of hardware sharing is described in NIC 7130. It is a protocol for the use of the LDS-1 processor at M.I.T. by anyone on the net who has a program for the LDS-1. This Graphics Loader, as it is called, provides for the execution of programs that have been sent to the PDP-10 at M.I.T. and the return of the data generated when the program is executed. The picture is

not drawn on a display, but since the LDS-1 processor can be instructed as to what to do with the coordinates that it generates, the Graphics Loader sets up the processor to write back into core the computed display coordinates. These coordinates may now be sent back to the originating site for display or as a debugging aid.

In NIC 7137 many of these previously discussed points are again brought up, but this time under the supposition that a graphics terminal should be just another terminal with minimal special privileges. Suggestions were also made pertaining to the design of a graphics protocol with particular emphasis on display structure, attention handling, coordinate systems, and the difference between storage tube and refreshed display requirements.

A specific suggestion for the handling of tablet input data has been presented, (NIC 7151), along with the expression that the graphics protocol should be designed so that non-interactive graphics should not be complicated with the requirements imposed by the interactive aspects of the protocol. It is pointed out that there are several types of tablet data that can be sent as input to a graphics process. Four types of data are described. They are single-shot, raw asynchronous, raw synchronous, and

preprocessed data. Preprocessed data can be smoothed or filtered or thinned using various techniques to make the data more uniform and workable, velocities can also be calculated for each point to aid in the interpretatin of the data.

The description of NETCRT (NIC 7172) is the first encounter with local processing, or lack of it. NETCRT is a protocol between a central processor and a character display. The character display is completely slaved to the central processor and can do no local processing, however it can interrupt the processor thus signalling that the user is done typing or wishes to begin typing. NETCRT tries to maintain good man-machine interaction by controlling the state of the terminal.

I have refrained from commenting on the various proposals as I summerized them because I don't think that it would have been in line with what I am trying to do in this paper. I think that there is a need to consider an overall model of the graphic system we are trying to design. Previous proposals have delt with some set of details without identifying with a general model, producing good ideas for implementation of details but not considering how the whole will fit together. Thus I would like to propose a model for our graphics system, it will contain many protocols and leave many areas to be discussed further, but it

will provide a starting point from which work can be done along simple lines, and yet not exclude the later inclusion of more sophisticated abilities.

Figure 1 shows a block diagram of information flow. The PROCESS indicates a graphics aplication program which is running on a computer in the net. Its associated INPUT and OUTPUT routines can be thought of as being a set of subroutines loaded with the main PROCESS or as separate and running elsewhere serving many users. At the other end of the loop are a set of INPUT and OUTPUT drivers for the DISPLAY which is being used to display the graphics information. The information flowing from the PROCESS to the DISPLAY is drawing information for the building and manipulating of pictures. The information flowing from the display to the PROCESS is attention information. The Graphic Data Base associated with the main PROCESS is that which is constructed when the picture is being drawn by the PROCESS or when the picture is being drawn by local processing and attention messages tell the PROCESS what has been done to the picture. This data base need not contain more information that the PROCESS is willing to work with, and in fact need not contain anything if no picture interaction is to be done. The Graphic Data Base associated with the DISPLAY drivers is built by themselves so that the OUTPUT driver can handle attentions from the DISPLAY

without requiring the main PROCESS to be able to do this and for the INPUT driver to use when modifying the picture based on what is actually being displayed. The information flowing to and from the main PROCESS is the sort which is passed or received as parameters to procedures. The INPUT and OUTPUT routines translates to and from respectively a network standard graphics protocol which is sent out over the net to the INPUT and OUTPUT display drivers whose responsibility it is to translate the standard message into the appropriate byte stream to drive the DISPLAY or translate the attention from the DISPLAY into a network standard message. The DISPLAY is assumed to handle its own refreshing if it requires any, so that there will be as little apparent difference between refreshed and non-refreshed displays as is possible.

This model provides for both simplicity of use for those doing simple things and power which is needed for those doing sophisticated interactive graphics. It can be used with a minimum of effort and overhead by setting runtime conditions to indicate that no interactive graphics will be done and all associated processing should be skipped, while still enabling other PROCESSes to do high powered graphics without going to a completely different set of routines and rules.

Due to the existance of two separate data bases, which must be kept up-to-data with each other there are two modes of operating this model. For lack of better names let us call them PROGRAM graphics and LOCAL graphios. The former indicates that the picture being displayed is constructed by the main PROCESS and all input from the user at the display is sollisited, thus the DISPLAY data base is only updated after and as a result of action by the main PROCESS. The latter indicates that the user at the display is directing the construction of a picture by means of function buttons and drawing tools, the DISPLAY data base is updated immediately and the main PROCESS is notified of the change so that it may keep up, but it does not perform manipulations of the picture unless requested to do so by the DISPLAY OUTPUT driver, this can be as a result of a request to perform some function that the DISPLAY INPUT/OUTPUT drivers can do by themseves or a request by the user to have the main PROCESS perform a non-standard function on the picture.

The main purpose of this design is to allow greatest generality of graphic configurations rather than minimum response time. The design for an optimum requires more exact specification of the hardware configuration and the proposed usage. Since neither of these variables can be known, and in fact our attempt at generality keeps us for even guessing very

closely at them, we must provide intelligent INPUT/OUTPUT drivers that will know how to split the processing load between themselves and the main PROCESS as a function of what kind of DISPLAY they are driving, rather than attempting to design in an optimum breakpoint.

The Graphics Protocol should specify the format of the messages which are transmitted between the INPUT and OUTPUT routines and drivers. These messages can be divided as previously mentioned according to their direction and content, i.e. drawing messages and attention messages. Since it is often desired to intermix graphics and text there should be a distinguishing message header for all graphics messages. Then a byte to specify the type of information contained in the body of the message, a count of the bytes in the body, and finally the body itself. Virtually all of the necessary message types have been indicated in the previous RFCs and I will not list them here, except to note that attentions now include requests for processing that the drivers could not do.

To summerize, I believe that a simple model is enough to enable the design of both sohisticated interactive graphics and low effort non-interactive graphics. The primary reason for this is that our major interest is not minimum response, but rather

maximum configuration mixes, There are opportunities to use software sharing and data reconfiguration services when building INPUT/OUTPUT routines and drivers. Much detailed work remains to be done, but with a basic model in sight providing a framework to hang proposed ideas on for evaluation, work should be able to proceed more smoothly.

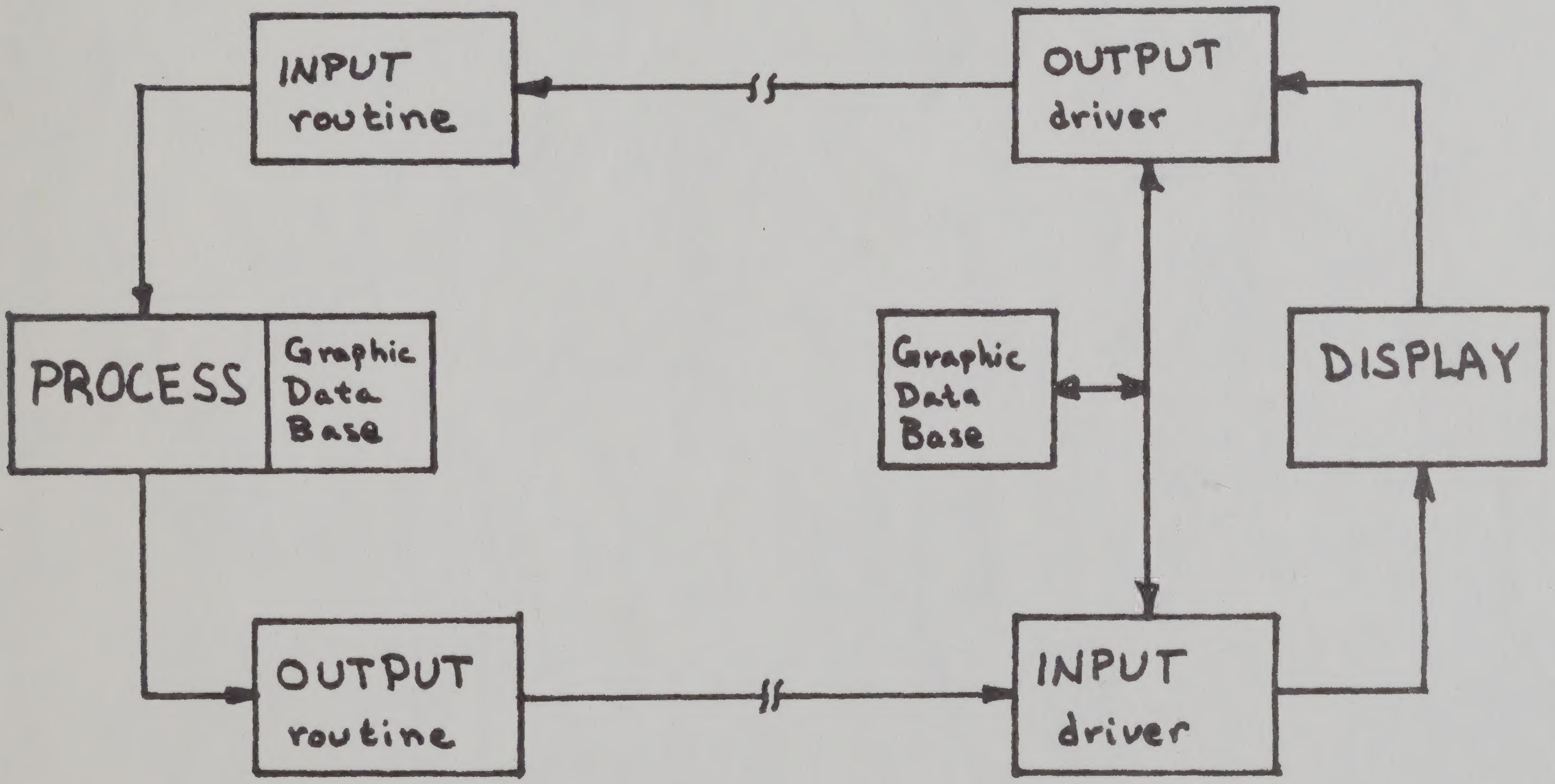
INPUT
routine
OUTPUT
driver
PROCESS
Graphic
Data
Base
Graphic
Data
Base
DISPLAY
OUTPUT
routine
INPUT
driver

Figure 1